E5994

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-154138
(43)Date of publication of application : 08.06.1999

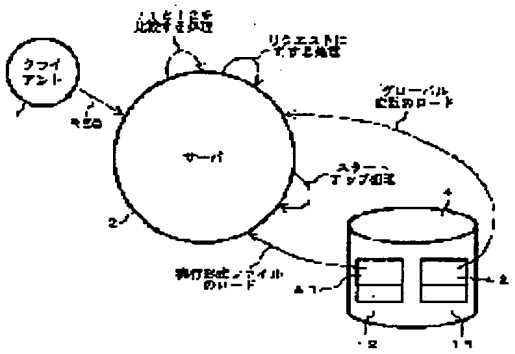
(51)Int.Cl. G06F 15/00
G06F 9/06
G06F 13/00

(21)Application number : 09-337673
(22)Date of filing : 21.11.1997

(71)Applicant : FUJI XEROX CO LTD
(72)Inventor : ASANO SADAJI

(54) SERVER

(57)Abstract:
PROBLEM TO BE SOLVED: To safely enable version-up of a module without having to stop a system concerning a client/server system in a distributed object environment.
SOLUTION: A global parameter and its structure information are backed up in a backup file 42 on a disk 4. The structure information of the global parameter on a module 41 is written out in the load module 41 on the disk 4 as well. At the time of start-up, two pieces of said structure information are compared. When both pieces of structure information are coincident as a result of comparison, start-up is made successful, When both pieces of structure information are not coincident by the version-up of the module, on the other hand, fail information is reported to a client or according to the structure information on the side of the module 41, the global parameter is defined again.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2000 Japanese Patent Office

This Page Blank (uspto)

E5994

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平11－154138

(43)公開日　平成11年(1999)６月８日

(51)Int.Cl.[6]		識別記号	FI		
Ｇ０６Ｆ	15/00	３１０	Ｇ０６Ｆ	15/00	３１０Ｃ
	9/06	４１０		9/06	４１０Ｑ
	13/00	３５７		13/00	３５７Ｚ

審査請求　未請求　請求項の数４　ＦＤ　（全６頁）

(21)出願番号　特願平9－337673

(22)出願日　平成９年(1997)11月21日

(71)出願人　000005496
富士ゼロックス株式会社
東京都港区赤坂二丁目17番22号

(72)発明者　浅野　貞二
埼玉県岩槻市府内３丁目７番１号　富士ゼロックス株式会社内

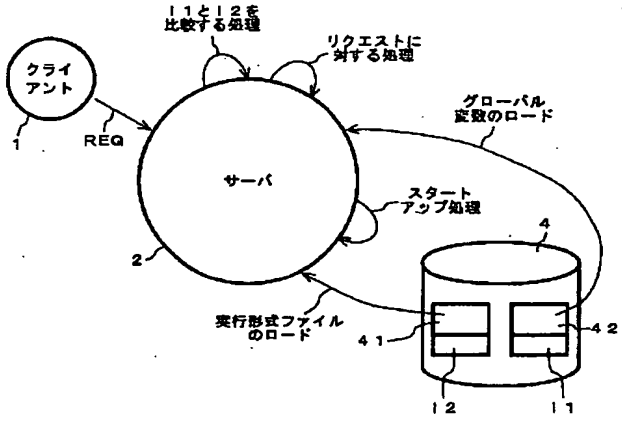
(74)代理人　弁理士　田中　香樹　（外１名）

(54)【発明の名称】　サーバ

(57)【要約】

【課題】　分散オブジェクト環境下におけるクライアント・サーバ・システムにおいて、システムを停止させることなく安全にモジュールのバージョンアップをすることができるようにする。

【解決手段】　ディスク４上のバックアップファイル４２にグローバル変数およびその構造情報をバックアップする。ディスク４上のロードモジュール４１にも、該モジュール上のグローバル変数の構造情報を書き出す。スタートアップ時に、前記二つの構造情報を比較する。比較の結果、両者が一致したときはスタートアップは成功する。一方、モジュールがバージョンアップされていて両者が不一致であったならば、クライアントにフェール情報を通知するか、モジュール４１側の構造情報に従ってグローバル変数を再定義する。



【特許請求の範囲】
【請求項１】　クライアント・サーバ型の分散オブジェクト環境に配置されたサーバにおいて、
サーバ上のグローバル変数の複製およびその構造情報（第１構造情報）を格納したバックアップファイルと、サーバのアプリケーション・プログラム、ならびに該アプリケーション・プログラム上のグローバル変数の構造情報（第２構造情報）を格納した実行形式ファイルと、サーバプロセスのスタートアップ時に、前記第１構造情報および第２構造情報の一致・不一致を判別するグローバル変数の構造比較手段と、
前記第１構造情報および第２構造情報が不一致のときにスタートアップ失敗を表すフェール情報をクライアントに通知するフェール処理手段とを具備したことを特徴とするサーバ。
【請求項２】　グローバル変数の位置の変更により前記不一致が生じた場合に、該グローバル変数を前記第２構造情報に従って再配置する再配置手段を具備したことを特徴とする請求項１記載のサーバ。
【請求項３】　グローバル変数の新たな追加により前記不一致が生じた場合に、該追加されたグローバル変数を初期化する初期化手段を具備したことを特徴とする請求項１記載のサーバ。
【請求項４】　グローバル変数間の依存関係が新たに発生して前記不一致が生じた場合に、該依存関係にあるすべてのグローバル変数を初期化する初期化手段を具備したことを特徴とする請求項１記載のサーバ。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、サーバに関し、特に、分散オブジェクト環境下におけるサーバプログラムのアップデート（更新）前後においてグローバル変数が整合していない場合にグローバル変数を再構成して処理を継続することができるサーバに関する。
【０００２】
【従来の技術】クライアント・サーバ型の分散オブジェクト環境下では、一般的にネットワーク上に存在する複数のホストコンピュータがクライアント・サーバ型のコンピュータシステムを構成している。このコンピュータシステムでは、各サーバは特定のホストコンピュータに依存することなく配置されていて、クライアントは目的のサーバ処理を実行するサーバが存在するホストコンピュータを意識することなく処理を依頼（リクエスト）することができる。このようなシステムでは、多数のコンピュータに分散されたオブジェクトが、代表的には、Common Object Request Broker Architecture（ＣＯＲＢＡ標準）によって統合的に管理されている。ＣＯＲＢＡ標準の環境下では、Object Request Broker（ＯＲＢ）と呼ばれるシステムプログラムが各ホストコンピュータに配置されていて、クライアントとサーバ間のやりとりを管理している。
【０００３】クライアントとサーバ間のインタフェースはInterface Difinition Language（ＩＤＬ）と呼ばれる記述言語であらかじめ定義されていなければならない。この記述言語で定義されたインタフェースは、すべてのホストコンピュータ上のＯＲＢが解釈可能であって、この定義に従ってクライアントからサーバへのリクエストが管理される。特開平８－２７２７２５号公報には、ＣＯＲＢＡ標準による分散オブジェクト環境においてサーバの構成情報を判定し、操作するためのシステムおよび方法が開示されている。
【０００４】上記コンピュータシステムにおいては、システム上に存在するすべてのサーバが、クライアントからのリクエストに即座に対応できるように常にアクティブ状態におかれていることが望ましい。しかし、このようにすべてのサーバを常にアクティブ状態に維持することは、コンピュータ上の資源を無駄に消費することとなる。
【０００５】そこで、クライアントからリクエストがない場合は、ＯＲＢの管理の下、サーバのプロセスを停止したシャットダウン状態にしている。すなわち、メモリやテーブル類等の資源を解放してプロセス・コントロール・ブロックを消去している。
【０００６】そして、新たなリクエストが発行されたときに、ＯＲＢは新たなプロセスをサーバにアサインし（割り当て）、サーバをアクティブ状態にする。アクティブ状態になると、サーバの記憶空間はシャットダウン前とは別の新たな記憶空間となり、先に記憶空間に存在したカウンタ値等のグローバル変数の値は保証されない。
【０００７】サーバは、このような不具合に鑑み、ディスク上のファイルにグローバル変数の複製をもち、シャットダウン後のスタートアップ時に、この複製のグローバル変数を読み出すように構成している。このグローバル変数のバックアップ方式は、システムのクラッシュ、例えば、バグ等によるプロセス停止の後の復旧時にもグローバル変数のバックアップ方式として利用することができる。
【０００８】分散オブジェクト環境下では、特定のサーバの保守をするときにもシステム全体を停止しなくてもよいという利点がある。例えば、モジュールつまりアプリケーション・ファイルのアップデートの際、サーバがシャットダウン状態においてはプロセスとサーバとの対応関係がなくなっていることから、ディスク上のサーバの実行形式ファイルつまりサーバのアプリケーション・プログラムを入れ替えるのみで、新たなリクエストを受けたとき、サーバはアクティブ状態になる。
【０００９】
【発明が解決しようとする課題】しかし、上記バックアップされたグローバル変数を使用して新たなリクエスト

に応答する方式では、ディスク上のファイルにバックアップされたグローバル変数の構造に変更がない場合にのみ正常な起動が可能である。すなわち、モジュールのアップデートの際、新しいサーバでグローバル変数の構造がシャットダウン前から変更されている場合には、ディスク上のサーバの実行形式ファイルを入れ替えるのみでは、サーバをアクティブ状態にすることができない。

【００１０】本発明は、上記問題点を解消し、モジュールのアップデートの際、シャットダウン前後においてグローバル変数の変更があった場合でも容易にスタートアップさせることができるサーバを提供することを目的とする。

【００１１】

【課題を解決するための手段】上記の課題を解決し、目的を達成するための本発明は、クライアント・サーバ型の分散オブジェクト環境に配置されたサーバにおいて、サーバ上のグローバル変数の複製およびその構造情報（第１構造情報）を格納したバックアップファイルと、サーバのアプリケーション・プログラム、ならびに該アプリケーション・プログラム上のグローバル変数の構造情報（第２構造情報）を格納した実行形式ファイルと、サーバプロセスのスタートアップ時に、前記第１構造情報および第２構造情報の一致・不一致を判別するグローバル変数の構造比較手段と、前記第１構造情報および第２構造情報が不一致のときにスタートアップ失敗を表すフェール情報をクライアントに通知するフェール処理手段とを具備した点に特徴がある。

【００１２】上記特徴によれば、実行形式ファイルにアプリケーション・プログラムのグローバル変数の構造情報をもたせた。したがって、アプリケーション・プログラムのアップデートでグローバル変数の構造に変更があった場合に、そのことを、第１および第２構造情報の比較によって判別することができる。そして、グローバル変数の変更があった場合は、そのままスタートアップできないので、クライアントへスタートアップ失敗を通知してオペレータの介入を促すことができる。

【００１３】

【発明の実施の形態】以下、本発明の実施形態を図面を参照して詳細に説明する。図２は、本発明の一実施例に係るクライアント・サーバ・システムの構成を示すブロック図である。同図において、ＬＡＮ等のネットワーク上にワークステーションＷＳ１，ＷＳ２，ＷＳ３が接続されている。ワークステーションＷＳ１，ＷＳ２，ＷＳ３は、それぞれシステムプログラムとしてＯＲＢを有している。ここでは、ワークステーションＷＳ１にクライアント１が存在し、ワークステーションＷＳ２，ＷＳ３にサーバ２，３が存在する場合を想定している。

【００１４】図３は、上記クライアント・サーバ・システムのインタフェースの構造を示す図である。同図において、ワークステーションＷＳ１～ＷＳ３はシステムプログラムとしてのＯＲＢと記述言語ＩＤＬで定義されたインタフェースを介してデータの授受をする。すなわち、ワークステーションＷＳ１上のクライアント１で発行されたリクエストＲＥＱはＩＤＬおよびＯＲＢを介してワークステーションＷＳ２，ＷＳ３上のサーバ２，３に通知される。また、前記各サーバ２，３での処理結果は、ＩＤＬ５およびＯＲＢ６を介して前記クライアント１に通知される。

【００１５】クライアント１がサーバ２に対してリクエストを発した場合のサーバ２の動作を図１を参照して説明する。サーバ２はコンピュータとこのコンピュータ上で実行されるアプリケーション・プログラムをロードするためのメモリを有する。ディスク４はアプリケーション・プログラムを格納した実行形式ファイル４１とグローバル変数のバックアップファイル４２とを有している。グローバル変数は、サーバ２でのサービスの実行によって変数の書き換えが発生する毎に、バックアップファイル４２に格納、つまりバックアップされる。したがって、シャットダウン時には、グローバル変数が確実にバックアップされており、再度のスタートアップ時には、バックアップファイル４２のグローバル変数を使用することができる。なお、バックアップファイル４２にはクローバル変数のほか該変数の構造情報Ｉ１も格納されている。前記実行形式ファイル４１もアプリケーション・プログラム上のグローバル変数の構造情報Ｉ２を有している。

【００１６】サーバ２はリクエストＲＥＱを受けるまではシャットダウン状態である。サーバ２はリクエストＲＥＱを受けると、ＯＲＢの管理動作によりディスク４上の実行形式ファイル４１をサーバ３のメモリ上にロードする。実行形式ファイル４１がロードされると、サーバ２は自己のスタートアップ処理を開始する。スタートアップ処理では、例えばローカル変数や資源つまりメモリやＩ／Ｏの初期化が行われる。スタートアップ処理には、ディスク４上のバックアップファイル４２に格納されているグローバル変数の複製をロードして、シャットダウン前の状態の復元を図る処理も含まれる。

【００１７】サーバ２における復元処理では、必要に応じてグローバル変数の再配置が実行される。すなわち、シャットダウン状態の間にアプリケーション・プログラムがアップデートされている場合がある。この場合、シャットダウン前後でバックアップファイル４２のグローバル変数の構造情報Ｉ１とアプリケーション・プログラム上のグローバル変数の構造情報Ｉ２とが異なっていることがある。そこで、グローバル変数の再配置では、グローバル変数のバックアップファイル４２に含まれているグローバル変数の構造情報Ｉ１と実行形式ファイル４１に含まれているグローバル変数の構造情報Ｉ２とを比較する。比較の結果、両者が一致した場合はシャットダウン前の状態が復元されたので、クライアント１からの

リクエストＲＥＱの処理を開始する。

【００１８】一方、サーバ２がシャットダウン状態の間に前記実行形式ファイル４１がアップデートされていて両者が一致しない場合は、グローバル変数の補正処理を行った後にリクエストＲＥＱの処理を開始する。

【００１９】グローバル変数の構造情報Ｉ１，Ｉ２が一致しない場合、その不一致の態様により復元処理が異なる。以下、グローバル変数のディスク４上の位置が異なる場合と、新たなグローバル変数が追加された場合を説明する。図４（ａ）はバックアップされていたグローバル変数の構造情報Ｉ１の一例であり、図４（ｂ）は実行形式ファイル４１に含まれるグローバル変数の構造情報Ｉ２の一例である。構造情報Ｉ１，Ｉ２には、各グローバル変数のタイプ、サイズ、位置、および依存関係が含まれている。

【００２０】図４の構造情報Ｉ１，Ｉ２から、シャットダウン前後では変数Ａおよび変数Ｃの位置が変更されているのに加え、変数Ｂが追加されていることがわかる。さらに、追加された変数Ｂは変数に対してプログラム上で依存関係にあることも記述されている。依存関係の例としては、例えば、変数Ａを算出するときに、変数Ｂが用いられる場合がある。したがって、このような依存関係があるときには、変数Ａが初期化される時には変数Ｂも同時に初期化される必要がある。

【００２１】なお、依存関係を含む構造情報は、実行形式ファイル４１をコンパイルするコンパイラの拡張機能により生成するように構成できる。例えば、アルゴリズム上、「Ａ．ｔａｂｌｅ＝Ｋ＋Ｂ；　Ｉｆ　Ａ．ｔａｂｌｅ＞４　ｔｈｅｎ　Ａ．ｓｅａｔ＝ＴＲＵＥ」となっていれば、コンパイラの構文解析機能により図４（ｂ）に示した依存関係が生成される。

【００２２】シャットダウン前後の構造情報が異なるグローバル変数の復元処理をフローチャートを参照して説明する。図５において、スタートアップされた後、ステップＳ１では前記構造情報Ｉ１とＩ２とを比較する。ステップＳ２では、前記構造情報Ｉ１とＩ２とが一致しているか否かを判断する。一致していなかった場合は、ステップＳ３に進み、追加されているグローバル変数があるか否かを判断する。グローバル変数が追加されていた場合はステップＳ４に進み、追加されているグローバル変数を初期化する。この実施形態では変数Ｂが追加されていて該変数Ｂはバックアップファイル４２からロードされることはないので、該変数Ｂの初期化が行われる。グローバル変数の初期化は、該変数Ｂのオブジェクトにおけるコンストラクタを有効にすることによって実行される。例えば、変数Ｂがカウンタ値である場合、該カウンタ値のインクリメントおよびデクリメントを実行するオブジェクトにおいて、該オブジェクトの生成（インストール）時にコンストラクタが有効となり、グローバル変数であるカウンタ値に「０」がセットされ、初期化される。

【００２３】ステップＳ５では追加された変数が他のグローバル変数と依存関係にあるか否かを判断する。依存関係があればその依存関係にある変数を初期化する。この実施形態ではグローバル変数Ａの要素である「ｔａｂｌｅ」および「ｓｅａｔ」と依存関係があるので、これらの変数を初期化する。グローバル変数の追加がない場合、および追加があっても依存関係がない場合は、ステップＳ４～Ｓ６はスキップされる。

【００２４】ステップＳ７では、グローバル変数の位置に変更があったか否かを判断する。この判断が肯定ならばステップＳ８に進み、グローバル変数をディスク４上に再配置する。この実施形態では、構造情報Ｉ２に従って、グローバル変数を再構成する。すなわち、変数Ａとして位置（Ｌｏｃａｔｉｏｎ）１に置かれていた値は位置３に変数Ａとして再配置される。同様に、変数Ｃとして位置３に置かれていた値は位置１に変数Ｃとして再配置される。こうして再配置されたグローバル変数はサーバ２のメモリへロードされる。ステップＳ９では、クライアント１から要求があったサービスを開始する。グローバル変数の位置変更がなかった場合は、ステップＳ１０でグローバル変数の追加があったか否かを判断し、この判断が肯定ならば、復元処理が終了したので、ステップＳ９に進む。ステップＳ１０が否定ならば追加構造情報Ｉ１，Ｉ２の不一致はないので、ステップＳ１１に進んでフェール情報をクライアント１に通知してスタートアップ失敗で終了する。

【００２５】上述の実施形態では、変数の位置の変更および変数の追加があった例を示した。本発明は、これらに限定されず、変数のタイプや変数のサイズが変更されていた場合にも同様に、該当するグローバル変数を初期化することで対応できる。

【００２６】また、本実施形態では、グローバル変数の構造情報Ｉ１，Ｉ２が異なった場合、グローバル変数を再配置してロードする例を示した。しかし、これに限らず、グローバル変数の不一致を検出した場合、サービス開始のための処理を中断し、オペレータへの介入を要求するフェール情報をクライアント１に通知してもよい。これによってクライアント１がリカバリー処理に入ることができる。また、不一致となったグローバル変数が、あらかじめ定めた特定の変数である場合にのみ、前記フェール情報を通知するようにしてもよい。

【００２７】以上のように、本実施形態によれば、サーバプログラムつまりサーバのアプリケーション・プログラムのアップデートにより、グローバル変数の構造に変更が生じた場合は、クライアントに通知してオペレータの介入を促すことができる。また、サーバプログラムのアップデートによるグローバル変数の構造変更に応じて自動的にグローバル変数の再定義をすることができるので、システムを停止することなく安全に実行形式ファイ

ルの入れ替えをして、サーバプログラムのアップデートをすることができる。

【００２８】

【発明の効果】以上の説明から明らかなように、本発明によれば、サーバプログラムのバージョンアップ等によってグローバル変数の構造に変更が生じた場合、この構造の変更を判別して、クライアントに通知する等の適当なリカバリ処理をすることができる。

【図面の簡単な説明】

【図１】　本発明の一実施形態に係るクライアント・サーバメシステムのデータの流れを示す図である。

【図２】　本発明の一実施形態に係るクライアント・サーバ・システムの構成を示すブロック図である。

【図３】　分散オブジェクト環境下のクライアント・サーバ・システムのインタフェースの構造を示す図である。

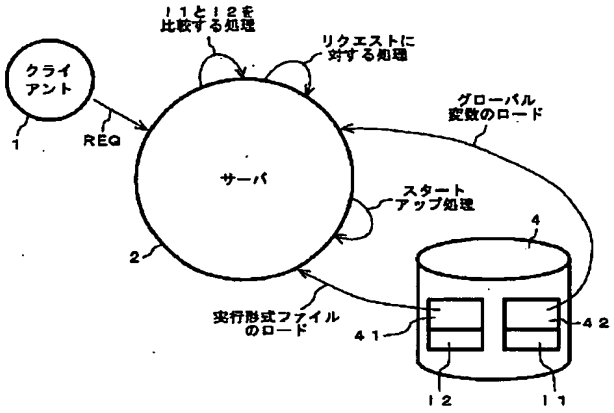
【図４】　グローバル変数の構造情報の一例を示す図である。

【図５】　グローバル変数の再定義のフローチャートである。

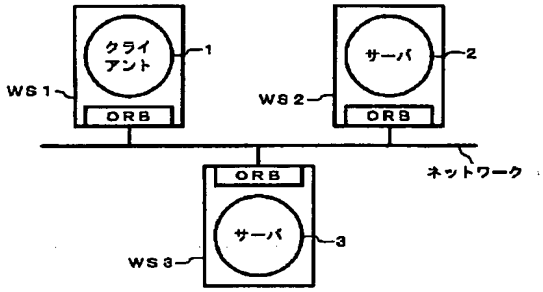
【符号の説明】

１…クライアント、　２，３…サーバ、　４…ディスク装置、　４１…実行形式ファイル、　４２…バックアップファイル、　Ｉ１，Ｉ２…構成情報

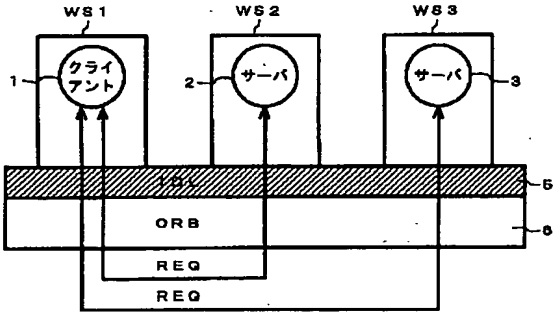
【図１】



【図２】



【図３】



【図４】

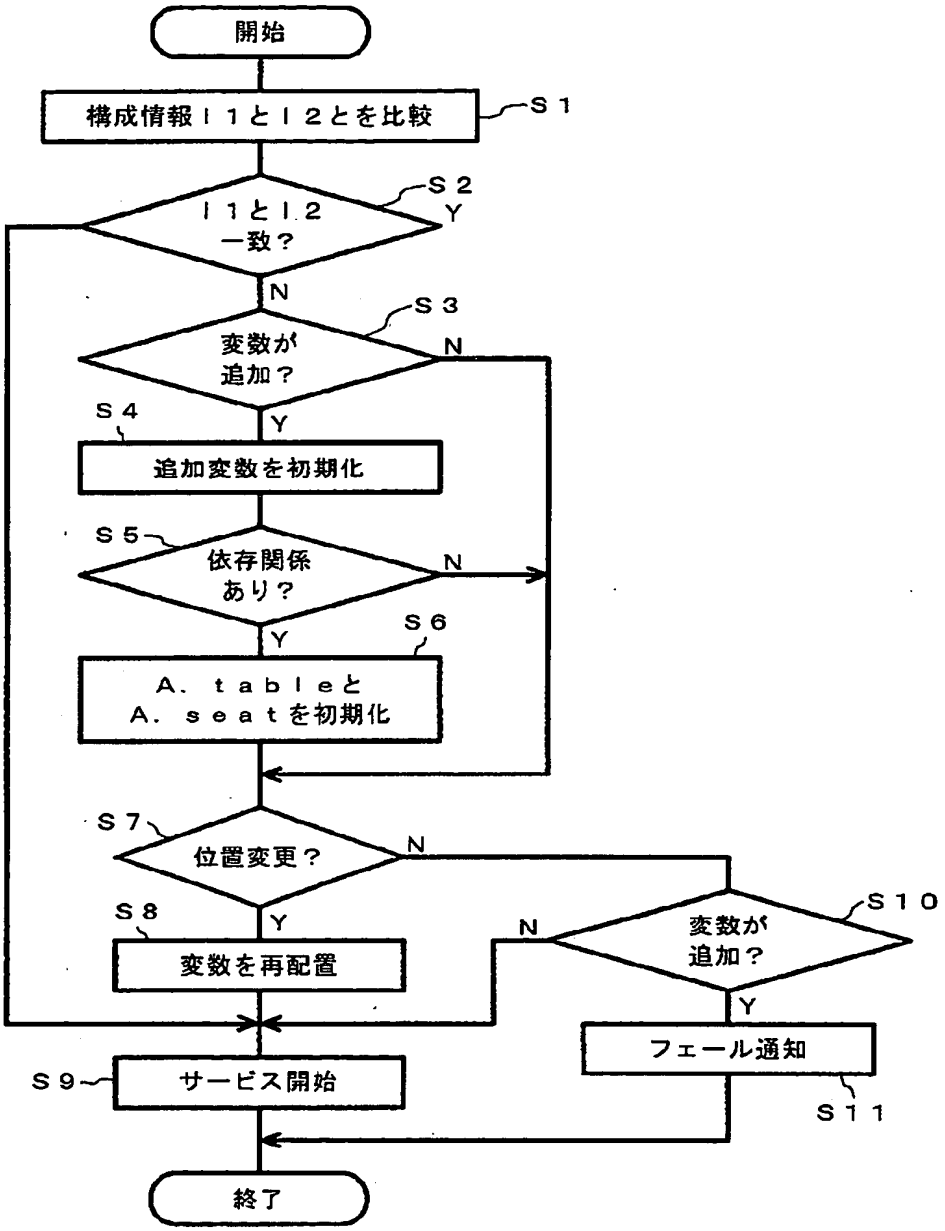
(a)

名称	タイプ	サイズ	位置	依存関係
A	A_struct	2435	Location1	
C	C_struct	200	Location3	

(b)

名称	タイプ	サイズ	位置	依存関係
A	A_struct	2435	Location3	
B	B_struct	3321	Location2	A.table, A.seat
C	C_struct	200	Location1	

【図５】


開始
構成情報Ｉ１とＩ２とを比較
Ｓ１
Ｉ１とＩ２一致？
Ｓ２
Ｙ
Ｎ
変数が追加？
Ｓ３
Ｎ
Ｙ
Ｓ４
追加変数を初期化
Ｓ５
依存関係あり？
Ｎ
Ｙ
Ｓ６
Ａ．ｔａｂｌｅとＡ．ｓｅａｔを初期化
Ｓ７
位置変更？
Ｎ
Ｓ８
Ｙ
変数を再配置
Ｎ
変数が追加？
Ｓ１０
Ｙ
フェール通知
Ｓ１１
Ｓ９
サービス開始
終了